PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-272717
(43)Date of publication of application : 18.10.1996

(51)Int.Cl. G06F 13/00
G06F 13/10

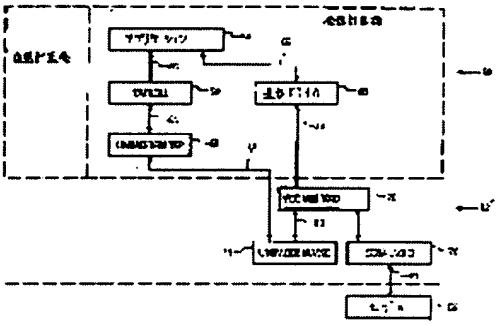
(21)Application number : 07-324647
(22)Date of filing : 13.12.1995

(71)Applicant : MICROSOFT CORP
(72)Inventor : WILLIAMS ROBERT J
GAPUTO CHRISTOPHER P
LAEPPLE KEITH A

(30)Priority
Priority number : 94 356059 Priority date : 13.12.1994 Priority country : US

(54) MODEM INTERFACE

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a modem interface which does not depend on a device.
SOLUTION: This interface uses call control for an application program and a data transmission application program interface and accesses a modem by a method which does not depend on a device. A general purpose modem driver 74 reads device specified information from a registry and generates a specified control command to a modem 56. The driver 74 divides modem transparency for the application 54 into command and data modes. This modem interface detects the modem 56 and contains support which loads the device specified information to a registry based on modem information.



LEGAL STATUS

[Date of request for examination] 13.12.2002
[Date of sending the examiner's decision of rejection] 05.09.2006
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（A）**　(11)特許出願公開番号

特開平8－272717

(43)公開日　平成８年(1996)10月18日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	13/00	３５３	7368－5E	Ｇ０６Ｆ	13/00	３５３Ｃ
	13/10	３３０	7368－5E		13/10	３３０Ｃ

審査請求　未請求　請求項の数19　ＯＬ　（全 16 頁）

(21)出願番号　特願平7－324647

(22)出願日　平成７年(1995)12月13日

(31)優先権主張番号　０８／３５６０５９

(32)優先日　1994年12月13日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　594115430
マイクロソフト　コーポレイション
Ｍｉｃｒｏｓｏｆｔ　Ｃｏｒｐｏｒａｔｉｏｎ
アメリカ合衆国　98053－6399　ワシントン州　レドマンド　ロケイション　８エス－2076　ワン　マイクロソフト　ウェイ（番地なし）

(74)代理人　弁理士　杉村　暁秀　（外４名）

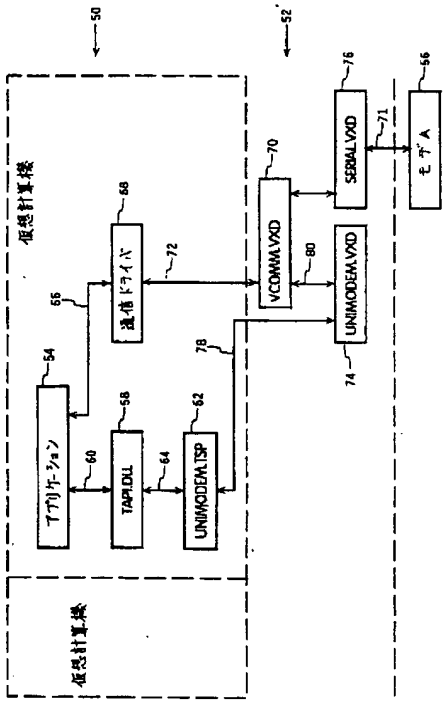
最終頁に続く

(54)【発明の名称】　モデムインタフェース

(57)【要約】

【課題】　デバイスに依存しないモデムインタフェースを提供する。

【解決方法】　本インタフェースは、アプリケーションプログラム用の呼制御およびデータ伝送アプリケーションプログラミングインタフェースを用い、デバイスに依存しない方法でモデムにアクセスする。汎用モデムドライバは、デバイス特定情報をレジストリから読み取り、特定制御コマンドをモデムに対して発生する。汎用モデムドライバは、アプリケーションに対するモデム透過性をコマンドモードとデータモードとに分離する。本モデムインタフェースは、モデムを検出し、デバイス特定情報をモデム情報に基づくレジストリにロードするサポートを含む。



1

【特許請求の範囲】

【請求項１】　１つまたはそれ以上のアプリケーションプログラムに対するデバイスに依存しないモデムインタフェースにおいて、

モデム特定制御情報を格納するレジストリデータ記憶装置と、

モデムに依存しない呼制御コマンドを受け、モデムを制御する呼コントローラと、

モデムに依存しないデータＩ／Ｏコマンドを受け、モデムを経てデータを転送する通信ドライバとを具え、

汎用モデムドライバが前記レジストリデータ記憶装置と通信してモデムに関するモデム特定制御情報を読み取り、前記汎用モデムドライバが前記呼コントローラと通信してモデムに依存しない呼制御コマンドを受けるとともに前記通信ドライバと通信してモデムに依存しないデータＩ／Ｏコマンドを受け、前記汎用モデムドライバが前記モデムと通信して前記モデムに依存しない呼制御コマンドを前記レジストリデータ記憶装置における前記モデム特定制御情報に基づくモデム特定制御コマンドに変換することを特徴とするデバイスに依存しないモデムインタフェース。

【請求項２】　請求項１に記載のデバイスに依存しないモデムインタフェースにおいて、前記汎用モデムドライバが、モデムのコマンドモードとデータモードとを区別し、モデムがコマンドモードの場合、前記汎用モデムドライバは、前記呼コントローラから呼制御コマンドをモデムに転送し、モデムがデータモードの場合、前記汎用モデムコントローラは、データＩ／Ｏコマンドをモデムに転送するように構成したことを特徴とするデバイスに依存しないモデムインタフェース。

【請求項３】　請求項１に記載のデバイスに依存しないモデムインタフェースにおいて、前記レジストリデータ記憶装置が、情報ファイルからモデム特定制御情報をこのレジストリにロードするクラスインストーラと通信することを特徴とするデバイスに依存しないモデムインタフェース。

【請求項４】　１つまたはそれ以上のポートドライバとデータ通信を行うポートインタフェースをさらに具える請求項１に記載のデバイスに依存しないモデムインタフェースにおいて、

前記汎用モデムドライバが、前記１つまたはそれ以上のポートドライバの１つとポートドライバの形式に依存しないで通信するポートドライバインタフェースと通信するように構成したデバイスに依存しないモデムインタフェース。

【請求項５】　請求項１に記載のデバイスに依存しないモデムインタフェースにおいて、前記呼コントローラおよび通信ドライバを一つのモジュールに結合し、１つまたはそれ以上のアプリケーションプログラムにアクセス可能なアプリケーションプログラミングインタフェース

2

を設けることを特徴とするデバイスに依存しないモデムインタフェース。

【請求項６】　請求項１に記載のデバイスに依存しないモデムインタフェースにおいて、前記呼コントローラおよび通信ドライバを別個のモジュールとし、各々が１つまたはそれ以上のアプリケーションプログラムにアクセス可能なアプリケーションプログラミングインタフェースを設けることを特徴とするデバイスに依存しないモデムインタフェース。

【請求項７】　請求項６に記載のデバイスに依存しないモデムインタフェースにおいて、前記呼コントローラを動的リンクライブラリとしたことを特徴とするデバイスに依存しないモデムインタフェース。

【請求項８】　請求項６に記載のデバイスに依存しないモデムインタフェースにおいて、前記通信ドライバを動的リンクライブラリとしたことを特徴とするデバイスに依存しないモデムインタフェース。

【請求項９】　請求項１に記載のデバイスに依存しないモデムインタフェースにおいて、モデム検出コマンドをモデムに発生し、モデム特定識別情報をモデムから受け、前記モデム特定識別情報に基づくデバイスＩＤを発生するクラスインストーラをさらに含むことを特徴とするデバイスに依存しないモデムインタフェース。

【請求項１０】　１つまたはそれ以上のアプリケーションプログラムにアクセスできる呼コントローラおよび通信ドライバと、モデム特定コマンドをモデムに転送する汎用モデムドライバと、モデム特定制御情報を格納するレジストリデータ記憶装置とを有するコンピュータシステムにおいて、モデムを経て通信する方法において、

前記汎用モデムドライバがモデム特定制御情報をモデムに発生できるように、汎用モデムドライバによってモデム特定制御情報を読み取ることと、

前記呼コントローラにおいて呼制御コマンドを受けることと、

前記通信ドライバにおいてデータＩ／Ｏコマンドを受けることと、

前記呼コントローラにおいて呼制御コマンドを受けたことに応じて呼制御コマンドを前記汎用モデムドライバに発生することと、

前記通信ドライバにおいてデータＩ／Ｏコマンドを受けたことに応じてデータＩ／Ｏコマンドを前記汎用モデムドライバに発生することと、

前記モデムがコマンドモードの場合、モデム特定制御コマンドを汎用モデムドライバによってこのモデムに転送することと、

前記モデムがデータモードの場合、データＩ／Ｏコマンドを汎用モデムドライバを経てこのモデムに転送することとを具えることを特徴とする方法。

【請求項１１】　請求項１０に記載の方法において、

前記レジストリデータ記憶装置と通信し、モデム情報フ

3

ァイルからモデム特定制御情報を前記レジストリデータ記憶装置にロードするクラスインストーラを設けるステップと、
前記モデムと通信し、このモデムの形式を検出するエニュメレータ検出モジュールを設けるステップと、
前記モデムが前記コンピュータシステムのバスに接続されている場合、前記エニュメレータ検出モジュールによってモデム形式を検出するステップと、
モデム形式の検出に応じて、前記クラスインストーラに対して前記モデム形式を識別するステップと、
前記クラスインストーラにおいて前記モデム形式を受けるのに応じて、前記クラスインストーラによってモデム特定制御情報を前記レジストリデータ記憶装置にロードするステップとをさらに含むことを特徴とする方法。

【請求項１２】　請求項１１に記載の方法において、前記検出ステップが、前記モデムからモデム識別情報を読み取ることを含み、前記識別ステップが、前記モデム識別情報に基づくデバイスＩＤを前記クラスインストーラに報告することを含むことを特徴とする方法。

【請求項１３】　請求項１０に記載の方法において、
前記モデムと通信し、モデム形式を検出するクラスインストーラを設けるステップと、
前記クラスインストーラによって質問の組を前記モデムに対して発生するステップと、
前記質問の組に対する前記モデムの応答を受けるステップと、
前記モデムの応答を、前記質問の組の発生された質問に依存してきれいにし、きれいにした応答を得るステップと、
前記きれいにした応答を個別デバイスＩＤに変換するステップと、
前記個別デバイスＩＤに適合するものを見つけようとするステップと、
前記個別デバイスＩＤに適合するものが見つかった場合、前記個別デバイスＩＤに対応するモデム特定制御情報を前記レジストリデータ記憶装置にロードするステップとをさらに含むことを特徴とする方法。

【請求項１４】　請求項１０に記載の方法において、
前記レジストリデータ記憶装置と通信するコンフィギュレーションマネージャと、前記汎用モデムドライバをロードするデバイスローダとを設けるステップと、
前記コンフィギュレーションマネージャによって前記レジストリデータ記憶装置を読み取り、前記デバイスローダを決定するステップと、
アプリケーションプログラムが前記呼制御モジュールにコマンドを発生し、前記モデムとの通信を確立する場合、前記デバイスローダによって補助記憶装置から前記汎用モデムドライバをメインメモリにロードするステップとをさらに含むことを特徴とする方法。

【請求項１５】　コンピュータシステムへの導入中にモデムを検出する方法において、

4

前記コンピュータシステムの通信ポートを経て第１の質問を発生し、モデムが装着されているかどうかを決定することを具え、
モデムが応答した場合、
前記モデムにモデム識別情報を送信させるように命令する質問を発生するステップと、
前記質問に対するモデムの応答を受けるステップと、
前記質問の発生された質問に依存して前記モデムの応答をきれいにして、きれいにした応答を得るステップと、
前記きれいにした応答をデバイス識別に変換するステップとを実行することを特徴とする方法。

【請求項１６】　請求項１５に記載の方法において、
前記コンピュータシステムに装着されたモデムについての情報を保持するレジストリデータ記憶装置と、各々がデバイス識別に関係し、モデム特定制御情報を含む情報ファイルとを設けるステップと、
前記変換ステップから得たデバイス識別に適合するものを見つけようとするステップと、
前記デバイス識別に適合するものが見つかった場合、前記デバイス識別に対応するモデム特定制御情報を前記レジストリデータ記憶装置にロードするステップとをさらに含むことを特徴とする方法。

【請求項１７】　請求項１５に記載の方法において、
前記デバイス識別に適合するものが見つからない場合、汎用デバイス識別を発生するステップをさらに含むことを特徴とする方法。

【請求項１８】　請求項１５に記載の方法において、前記変換ステップが、
前記きれいにした応答に巡回冗長検査を行うことを特徴とする方法。

【請求項１９】　請求項１５に記載の方法において、巡回冗長検査表を瞬時に計算することを特徴とする方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、モデムを使用するコンピュータ通信と、さらに特にコンピュータシステムにおけるモデムインタフェースとに関するものである。

【０００２】

【発明が解決しようとする課題】今日のパーソナルコンピュータにおけるアプリケーションプログラム（アプリケーション）の数の増加は、リモートデータ通信の特徴をサポートするモデムと相互に影響し合っているに違いない。モデムにアクセスするアプリケーションのいくつかの例としては、リモートネットワークアクセスや、掲示板サービスや、ファックスモデムや、電子メール（ｅメール）等を含む必要があるかもしれない。ワードプロセッサや表計算プログラムのようなアプリケーションさえも、例えば、モデムを制御して他のコンピュータにファイルを転送するソフトウェアを含むことができる。モ

5

デムへのアクセスが、より多くのアプリケーションの重要な特徴になったことにより、アプリケーション開発者は、これらのアプリケーションにおけるモデム通信をサポートする開発コードの問題に取り組まなければならなくなった。

【０００３】モデム制御のタスクは、アプリケーション開発を極めて複雑にする。モデムを制御するために、アプリケーションは、コンピュータに接続された個々のブランドおよびバージョンのモデムのコマンドセットを理解する必要がある。多くのモデムは、ジョージア州ノークロス（Norcross, Georgia）にあるヘイズマイクロコンピュータプロダクツ社（Hayes Microcomputer Products, Inc.）によって制定されたヘイズ標準ＡＴコマンドセット（Hayes Standard AT Command set）によるコマンドを使用する。このＡＴコマンドセットは、コンピュータ産業において良く知られ、広く使用されているモデムを制御し、設定するための多くのコマンドを含む。しかしながら、各々のブランドまたはバージョンのモデムは、いくつかのＡＴコマンドと新たな特徴をサポートする追加のコマンドとを含むコマンドの独自の組み合わせを独自に理解する。この結果、モデムにおいてサポートされるコマンドセットは、実際的に異なる恐れがある。

【０００４】事態をさらに複雑にすることは、The Comite Consultatif International deTelegraphie et Telephonie （ＣＣＩＴＴ）と、The International Telecommunications Union - Telecommunications Standardization Sector（ＩＴＵ－Ｔ）とによって制定された、モデムおよびファクシミリ伝送に関する多数のプロトコルが存在することである。アプリケーション開発者は、アプリケーションがモデムを正確に制御できるようにするために、モデムプロトコルに加えてＡＴコマンドにも精通しなければならない。

【０００５】モデム技術の進歩とともに、モデムの特徴の変化と歩調を合わせてアプリケーションを制作することは、アプリケーション開発者にとって大変困難なことになった。各々の新たな特徴は、しばしば、１つまたはそれ以上の新たなコマンドによってサポートされ、これらのコマンドは、モデムのブランドによって異なるかもしれない。異なるモデムの特徴の範囲に適合するアプリケーションの制作において、モデムのサポートに向けられたアプリケーションの部分は、極めて複雑になる恐れがある。このコードの開発は、明らかにアプリケーションの中心部分の機能の開発を減じる。たとえ、開発者が、現在のモデムの特徴をサポートする完全な仕事を行ったとしても、アプリケーションにおけるモデムサポートは、将来のバージョンのモデムのより進歩した特徴をサポートしないであろう。アプリケーションが市販された後、進歩した特徴に対するサポートを容易に追加する変更はできない。

【０００６】アプリケーション開発の複雑化に加えて、

6

多くのアプリケーションにおいてモデム制御を置くことは、資源コンテンション（resource contention）問題を発生させる。各々のアプリケーションがモデム制御のサポートを準備する場合、アプリケーションにモデム資源を割り当てることは、不可能ではないにしても困難である。これらの事情において代表的に、モデムの制御を最初に得たアプリケーションは、自発的に制御を止めるまで、制御を継続する。この結果、モデムを制御するアプリケーションは、他のアプリケーションが電話呼出しに関係することを不可能にする。この状況において、シリアルポートドライバに対するインタフェースは、せいぜいいくらかのコンテンション管理を行えるだけである。しかしながらこのコンテンション管理は、ある程度制限される。

【０００７】モデムサポートソフトウェアが存在することの追加の欠点は、これらがモデムをコンピュータシステムに導入することを困難にすることである。アプリケーションを２つ以上のモデム形式の制御に対して設定できるが、代表的に、個々のモデム形式との通信に関してアプリケーションを正確に設定するためのユーザからの追加の情報を必要とする。モデム特定情報の導入を自動的に行うことができるならば、モデムの導入は、ユーザに対してはるかに簡単になるであろう。

【０００８】モデム制御システムが存在することの制限および欠点を軽減するために、本発明は、デバイスに依存しないモデムインタフェースを提供する。本発明は、このモデムに対するデバイスに依存しないインタフェースを、コンピュータシステムにおいて動作しているアプリケーションに与える手段を含む。

【０００９】

【課題を解決するための手段】本発明の実施例によれば、本発明は、コンピュータオペレーティングシステムの一部として含まれるデバイスに依存しないモデムインタフェースを具える。このインタフェースは、呼制御コマンドを受ける呼制御モジュールと、アプリケーションからデータ伝送コマンドを受ける通信ドライバとを含む。アプリケーションは、呼制御コマンドおよびデータ伝送コマンドを、呼制御モジュールおよび通信ドライバに、各々、デバイスに依存しない方法で送る。汎用モデムドライバは、呼制御モジュールからの命令に応じて、デバイス特定コマンドをモデムに供給する。特定のモデムの特有の能力を理解するために、汎用モデムドライバは、レジストリ（registry）データ記憶装置からモデム特定情報を読み取る。この情報を使用すると、汎用モデムドライバは、多数のモデム形式のためのデバイス特定コマンドを使用することができる。

【００１０】汎用モデムドライバは、アプリケーションに対するモデム透過性をコマンドモードおよびデータモードに分離する。モデムがコマンドモードである場合、汎用モデムドライバは、呼制御コマンドをモデムに伝送

する。モデムかデータモードである場合、汎用モデムドライバは、通信ドライバからデータＩ／Ｏコマンドをモデムに伝送する。

【００１１】本発明は、追加の導入の特徴を提供する。本発明のある実施例は、デバイス特定情報をレジストリデータストア中にロードするクラスインストーラ（class installer）モジュールを含む。いくつかのモデムデバイスは、コンピュータシステムにおけるバスに関してロードされたエニュメレータ（enumerator）検出モジュールに対して自分自身を識別させる。導入中、デバイス識別を、クラスインストーラモジュールに移し、適切なデバイス情報をロードできるようにする。

【００１２】モデムが自分自身を識別させられない場合、本発明の他の実施例は、モデム形式を識別する方法を提供する。この方法は、デバイス特定情報が得られるように、モデムに対して一連の質問をすることと、次にモデムの応答をきれいにすることとを含む。応答データを、デバイス特定情報をロードするのに使用されるデバイス識別に変換することができる。

【００１３】本発明の個々の実施例は、コンピュータ用通信システムが存在することに関して、幾つかの利点を提供する。本発明は、アプリケーション開発者の仕事を、はるかに簡単にする。開発者は、広範囲に多様なデバイスと通信するための簡単な共通インタフェースを使用することができる。本発明のある実施例において、モデムの制御を複数のレベルに分割することができ、最上位レベルにおいて、制御および通信アプリケーションインタフェースは、デバイスに依存しないインタフェースをモデムに提供し、より下位のレベルにおいて、汎用モデムドライバは、高レベルコマンドを、特定の形式のモデム用のコマンドに変換することができ、最後により下位のレベルにおいて、ポートドライバは、モデムデータと、特定の形式のポート用のコマンドとを変換することができる。

【００１４】モデム制御のタスクをアプリケーションから取り除いた後、モデム制御インタフェースは、資源コンテンションをより効果的に行うことができる。

【００１５】本発明は、モデムを導入し、設定するユーザの仕事を、はるかに簡単にする。ユーザは、モデムをただ単にプラグ接続することができ、システムが、残りの面倒をみる。

【００１６】汎用モデムドライバは、異なった形式のモデム用の個々のドライバを組み込む必要性を取り除く。汎用モデムドライバとともに、モデム特定情報のみをロードする必要がある。モデム情報の組み込みを、ユーザが、どんな形式のドライバを組み込む必要があるのか、またはどうやってそれを組み込むのかを知る必要なしに、行うことができる。

【００１７】

【発明の実施の形態】図１は、本発明の実施例を実行するコンピュータシステム２０のブロック図である。コンピュータシステム２０は、その基本的な要素として、コンピュータ２２と、入力デバイス２４と、出力デバイス２６とを含む。

【００１８】一般にコンピュータ２２は、バス構造体３２を通じて連絡する、中央処理ユニット（ＣＰＵ）２８と、メモリシステム３０とを含む。ＣＰＵ２８は、計算を行う演算器（ＡＬＵ）３３と、データおよび命令を一時的に記憶するレジスタ３４と、アプリケーションまたはオペレーティングシステムのようなコンピュータプログラムからの命令に応じてコンピュータシステム２０の動作を制御する制御ユニット３６とを含む。

【００１９】一般にメモリシステム３０は、ランダムアクセスメモリ（ＲＡＭ）および読み出し専用メモリ（ＲＯＭ）半導体装置のような媒体の形式の高速メインメモリ３８と、フロッピーディスク、ハードディスク、テープ、ＣＤＲＯＭ等および光または磁気記録材料を使用する他の装置のような媒体の形式の２次記憶装置４０とを含む。メインメモリ３８は、コンピュータのオペレーティングシステムおよび現在動作しているアプリケーションプログラムのようなプログラムを記憶する。メインメモリ３８は、ディスプレイデバイスによって画像を表示するためのビデオディスプレイメモリも含む。

【００２０】代表的に、入力デバイス２４および出力デバイス２６を、バス構造体３２によってコンピュータ２２に接続された周辺デバイスとする。入力デバイス２４を、キーボード、モデム、ポインティングデバイス、ペンまたは入力データをコンピュータに供給する他のデバイスとしてもよい。出力デバイス２６を、ディスプレイデバイス、プリンタ、サウンドデバイスまたはコンピュータからの出力データを供給する他のデバイスとしてもよい。

【００２１】モデム５６は、入力および出力装置２４、２６の双方として機能することができる。出力デバイスとして、モデムは、プログラムされたＣＰＵの制御に従って、メモリからコマンドおよびデータを受けることができる。入力デバイスとして、モデム５６は、ファックスまたはコンピュータデータを受け、このデータをさらに処理または記憶するためにメモリに転送することができる。モデム５６は、データ、ファックスまたは音声呼出し、または一回の呼出しにおけるこれらの形式の組み合わせをサポートできる。コンピュータシステムを電話機に結合すると、コンピュータは、電話番号をダイヤルすることによって、音声呼出しを始めることができる。呼出し者は、ひとたび通信が確立すると、電話機の送受話器を通じて音声通信を続ける。音声およびデータ通信を切り換えるために、モデム５６は、音声通信を開始し、コンピュータ間の通信を確立することができる。モデムを通じた多くのモードの電話通信が可能であることを、当業者には理解されたい。

9

【００２２】モデムに対してコマンドおよびデータを物理的に転送する処理は、モデムの形式に依存して変化する。様々の標準モデム構成が存在し、本技術分野において良く知られている。これらの構成は、例えば、内蔵または外付けモデム、またはノートブック型コンピュータにおいて広く普及しているＰＣＭＣＩＡ（パーソナルコンピュータメモリカードインターナショナルアソシエーション（Personal Computer Memory Card International Association））カード接続を含む。″内蔵″モデムは、インダストリースタンダードアーキテクチャ（Industry StandardArchitecture（ＩＳＡ））バス、エキスパンデッドインダストリースタンダードアーキテクチャ（Expanded Industry Standard Architecture（ＥＩＳＡ））バス、またはマイクロチャネルアーキテクチャ（Micro Channel Architecture（ＭＣＡ））バスのようなバス構造体３２を経てコンピュータに結合される。モデムを、ＰＣＭＣＩＡポートを経てコンピュータのバス構造体３２に結合することもできる。″外付け″モデムは、ＲＳ－２３２シリアルポートまたはパラレルポートを経てコンピュータのバス構造体３２に結合する。追加の例としては、モデムを、ＰＣＩ標準（Peripheral Component Interconnect 、Peripheral ComponentInterconnect Special Interest Groupによって支持されるローカルバス標準）に準拠したローカルバス構造体に結合してもよい。多くの他の変形例が可能であり、当業者にはよく知られている。

【００２３】図１は、いろいろな用途のあるコンピュータシステムの基本的な要素を説明するブロック図であり、本図は、コンピュータシステム２０の特別なアーキテクチャを説明することを意図していないことを理解されたい。例えば、コンピュータ設計の分野において既知の種々のバス構造体を、多くの方法において、望んだようにコンピュータシステムの要素の相互接続に使用できるため、個々のバス構造体を示していない。ＣＰＵ２８を、別個のＡＬＵ３３、レジスタ３４および制御ユニット３６によって構成してもよいし、マイクロプロセッサにおけるように、これらのＣＰＵの部品を一緒に集積した１つのデバイスとしてもよい。さらに、コンピュータシステムの要素の数および配置を、当技術分野において既知の方法（例えば、多数プロセッサ、クライアントーサーバシステム、コンピュータネットワーキング、等）において示され、記述されているものから変えてもよい。

【００２４】図２は、コンピュータシステム内の、本発明の実施例のアーキテクチャを説明するブロック図である。このアーキテクチャは、通信アプリケーションにコンピュータシステム２０に導入されたモデムへのハードウェアに依存しないアクセスを与えるモデムサブシステムを示す。この図は、ワシントン州レッドモンド（Redmond, Washington）のマイクロソフト社（Microsoft Co

10

rporation）によって開発されたWindows 95オペレーティングシステムにおけるモデムサブシステムの特別なアーキテクチャを示す。Windows 95オペレーティングシステムのアーキテクチャにおいて、DOS アプリケーションのような仮想計算機が属するリング３（５０）と仮想デバイスが属するリング０（５２）とを含む多数のレベルが存在する。この図は、Windows 95環境におけるモデムサブシステムの特別な例を説明するが、本発明は、この特別な実現に限定されるものではないことを理解されたい。例えば、モデムサブシステムのアーキテクチャを、オペレーティングシステムが設計されているＣＰＵの形式に依存して変えることができる。モデムサブシステムの構造に対する多くの変形例が、本発明の範囲から逸脱することなしに可能である。

【００２５】モデムサブシステムは、モデム５６へのアクセスを必要とする１つまたはそれ以上の通信アプリケーション５４に対するサポートを提供する。通信アプリケーション５４は、リモートネットワークアクセス、ファックスアプリケーション、ｅメールアプリケーション、掲示板サービスアプリケーション、等を含む。これらおよび他の種々のアプリケーションは、モデムへのアクセスを必要とし、モデムを制御し、モデムを経てデータＩ／Ｏを行う。

【００２６】テレフォニアプリケーションプログラミングインタフェース（Telephony Application Programming Interface（ＴＡＰＩ））モジュール５８（または、″呼コントローラ″）は、モデム５６を制御するアプリケーションにデバイスに依存しないインタフェースを提供する。この実現において、ＴＡＰＩモジュール５８を、動的にリンクされたライブラリ（dynamic linked library（ＤＬＬ））とする。ＴＡＰＩ６０を経てアプリケーション５４によって呼ばれると、ＴＡＰＩモジュール５８は、国際呼出しのダイアル、ある形式の国内呼出しに関する待機、国内呼出しの応答、現在の呼出しの停止、等のようなモデム制御機能をサポートする。

【００２７】ＴＡＰＩモジュール５８は、アプリケーション機能呼び出しを、デバイス特定制御を行うサービスプロバイダ６２に送る。このモデムサブシステムにおいて、サービスプロバイダを、ユニモデムサービスプロバイダ（unimodem service provider）６２とする。

【００２８】ユニモデムサービスプロバイダ（unimdm.tsp）６２は、ＴＡＰＩモジュール５８によってテレフォニサービスプロバイダインタフェース６４を経て呼び出された場合、モデムデバイスの制御を行うＤＬＬ実現機能でもある。これらのモデム制御機能は、ダイアル、応答、停止等を含む。ユニモデムサービスプロバイダ６２に、アプリケーションによって呼び出し、ユーザがモデムパラメータを設定するのを許可するダイアログウィンドウを表示させてもよい。

【００２９】データＩ／Ｏを呼制御ＡＰＩに組み込むこ

11

とができたとしても、このモデムサブシステムは、この目的のために分離した通信インタフェースを提供する。Windows 95オペレーションシステムにおいて、アプリケーションは、通信アプリケーションプログラミングインタフェース６６（ＣＯＭＭ　ＡＰＩ）に対して機能呼び出しを行い、モデム５６を設定し、このモデムを通じてのデータＩ／Ｏを行う。

【００３０】モデムサブシステムにおいてＣＯＭＭ　ＡＰＩ６６をサポートするモジュールは、通信ドライバ６８および仮想通信ドライバ（ＶＣＯＭＭ）７０を含む。通信ドライバ６８は、リング３レベルモジュールであり、アプリケーションが、通信デバイスをオープンし、通信デバイスから読み出し、通信デバイスに書き込むことを許可する機能を含む。通信ドライバ６８は、ＣＯＭＭ　ＡＰＩ６６をサポートし、通信アプリケーションに装置に依存しない方法でモデムを使用することを許可する。Windows 95オペレーティングシステムにおいて、通信ドライバは、Win32 カーネルの部分である。しかしながら、これは、本発明の変形例の１つに過ぎず、通信ドライバを、当業者には既知の種々の慣例的な方法において実現することができることを理解されたい。

【００３１】ＶＣＯＭＭ７０は、リング０レベルの仮想デバイスであり、システムにおける通信資源へのすべてのアクセスを管理する。これらの″通信資源″は、１つの非同期データストリームを供給する物理的または論理的なデバイスを含む。通信資源の例としては、シリアルポート、パラレルポート、およびモデムがある。

【００３２】通信ドライバ６８は、ＶＣＯＭＭによって供給される呼機能（７２）によって、ＶＣＯＭＭと通信する。ＶＣＯＭＭは、コンピュータシステムにおける通信資源に対するポートに依存しないインタフェースとして機能する。

【００３３】ＣＯＭＭ　ＡＰＩ６６は、データＩ／Ｏ機能、ユーザインタフェース機能、およびコンフィギュレーション（configuration ）機能を含む。アプリケーションは、データＩ／Ｏ機能を呼出し、モデムデバイス５６を通じてデータを送り、かつ受ける。ユーザインタフェース機能を経て、ユーザは、モデムの好適な動作特性を選択することができる。ダイアログボックスは、スピーカのオン／オフ、ボーレート、圧縮およびエラー訂正プロトコル、等のモデムパラメータを含むことができる。アプリケーションは、コンフィギュレーション機能を呼出し、モデム５６の好適な動作特性を決定または変更する。これらのコンフィギュレーション機能に応じて、通信ドライバ６８は、ＶＣＯＭＭを経てコンフィギュレーション機能呼び出しを汎用モデムデバイスドライバ（ユニモデムデバイスドライバまたはunimodem.vxd）７４に送る。

【００３４】ＶＣＯＭＭを、通信ドライバ６８およびユニモデムデバイスドライバ７４に対するポートに依存しないインタフェースとする。ＶＣＯＭＭ７０は、ポートドライバに対するアクセスを与え、通信資源におけるハードウェア動作を実行する。ポートドライバは、レジスタに対するＶＣＯＭＭ７０サービスをそれら自身に使用し、通信ハードウェアに対するアクセスを制御する。これらのサービスに加えて、ＶＣＯＭＭ７０は、クライアント仮想デバイスサービスを提供し、仮想デバイスが、ポートドライバが組み込まれているどのような通信資源も使用できるようにする。特に、ＶＣＯＭＭ７０は、通信資源のオープンおよびクローズと、通信資源の設定と、通信資源からの読み出しおよび通信資源への書き込みと、通信イベントの処理と、ポートドライバの拡張機能の呼び出しとを行うサービスを提供する。

12

【００３５】ポートドライバは、デバイス特定コードを含み、アプリケーションが、ポートに装着された物理的または論理的な特定のデバイスと通信することを可能にする。ポートドライバの一例を、図２に示すシリアルポートドライバ（serial.vxd）とする。シリアルポートドライバは、モデムサブシステムが、シリアルポート７１を経てモデム５６のようなシリアル通信デバイスと通信することを可能にする。このシリアルポートドライバ７６は、Windows オペレーティングシステムとともに設けられ、その動作は既知である。モデム５６を、多くの異なる構成においてコンピュータに結合できることから、他のポートドライバを使用してもよいことを理解されたい。この実現において、ポートドライバを、ブートタイムにおいてまたは要求されたときにメモリ内にロードする。

【００３６】ポートドライバを要求されてロードする一例として、Windows 95オペレーティングシステムにおけるプラグアンドプレイアーキテクチャ（Plug and Play Architecture）に準拠するポートドライバを、ドライバがサポートするポートをＶＣＯＭＭクライアントが開こうとする場合にロードする。プラグアンドプレイアーキテクチャは、マイクロソフト社によって開発されたWindows オペレーティングシステム環境のために設計された導入システムである。このアーキテクチャの関連した部分を、図３の導入サブシステムの参照とともに詳細に記述する。

【００３７】図２のモデム導入サブシステムにおいて、汎用モデムドライバは、ＴＡＰＩ６０からの呼制御機能およびＣＯＭＭ　ＡＰＩ６６からのデータ伝送機能の両方のためのインタフェースを提供する。汎用モデムドライバは、ユニモデムサービスプロバイダ６２と呼ばれるテレフォニサービスプロバイダと、ユニモデムデバイスドライバ（unimodem.vxd）７４と呼ばれるＶＣＯＭＭポートドライバとを含む。ユニモデムサービスプロバイダ６２は、ユニモデムデバイスドライバの保護モードインタフェースにおいて実行される機能（７８）に対する呼出しを行うことによって、ユニモデムデバイスドライバ

７４と通信する。ユニモデムサービスプロバイダ６２とユニモデムデバイスドライバ７４との間のインタフェースを、当業者に既知の慣例的な技術を使用する他の方法において実現してもよい。

【００３８】ユニモデムデバイスドライバは、モデム５６に対する直接呼出しは行わないが、ＶＣＯＭＭ７０に戻って呼出しを行い、モデム５６と通信する。ポートドライバに対するインタフェースとしてＶＣＯＭＭ７０を使用することによって、ユニモデムデバイスドライバ７０は、ポートに依存しなくなる。この実現において、示したポートは、シリアルポートドライバ（serial.vxd）７６であるが、ユニモデムデバイスドライバ７４は、いかなる形式のポートに結合されたモデム５６も制御できる。

【００３９】ユニモデムデバイスドライバ７４は、モデム５６とのすべての通信を制御する。この制御を実行するために、設定およびデータＩ／Ｏに加えて呼制御は、ユニモデムデバイスドライバ７４を通過する。呼制御のために、ユニモデムサービスプロバイダ６２は、ユニモデムデバイスドライバを呼び出し、ダイアル、応答、停止、等のようなモデム制御コマンドを起動する。ユニモデムサービスプロバイダ６２は、これらの制御呼出しを、インタフェース７８を使用するＶＣＯＭＭを経てユニモデムデバイスドライバ７４に渡す。これに応じて、ユニモデムデバイスドライバ７４は、これらの呼制御機能を、コンピュータシステム２０に結合されたモデム５６の形式に適合するフォーマットに変換する。ＶＣＯＭＭ７０に戻る呼びだし（８０）を行うことによって、ユニモデムデバイスドライバ７４は、モデムコマンドをシリアルポートドライバ７６にＶＣＯＭＭ７０を経てポートに依存しない方法において転送する。

【００４０】ＣＯＭＭ　ＡＰＩからのデータ転送機能に関して、ユニモデムデバイスドライバ７４は、データＩ／Ｏ呼出しを監視する。データＩ／Ｏ呼出しは、アプリケーション５４がＣＯＭＭ　ＡＰＩを呼出したときに発生する。通信ドライバ６８は、ＶＣＯＭＭ７０を経てこれらの呼出しをユニモデムデバイスドライバ７４に渡す。ユニモデムデバイスドライバ７４は、ＶＣＯＭＭ７０を経てこれらのコマンドを、モデム５６が装着されたポートドライバ７６に渡す。

【００４１】ユニモデムデバイスドライバ７４は、モデム５６の透過性をコマンドモードとデータモードとに分離する。モデム５６がコマンドモードにある場合、ユニモデムデバイスドライバ７４は、ポートドライバ７６を経てモデム制御コマンドをモデム５６に渡す。反対に、モデム５６がデータモードにある場合、ユニモデムデバイスドライバ７４は、ポートドライバ７６を経てデータＩ／Ｏコマンドをモデム５６に渡す。ユニモデムデバイスドライバ７４は、サービスプロバイダ６２およびアプリケーション５４に関連して、透過性をコマンドおよびデータモード間で切り換える。〝透過〟制御は、ユニモデムサービスプロバイダ６２が、データモード中にモデム５６にデータが送られているのを知らず、アプリケーション５４が、コマンドモード中にモデム５６に呼制御コマンドが送られているのを知らないことを意味する。

【００４２】モデムがコマンドモードの場合、設定および監視に関するＣＯＭＭ　ＡＰＩ機能が処理される。例えば、アプリケーション５６が、データの１０バイトを受けた場合そのことを知らせるためにＣＯＭＭ　ＡＰＩ６６において呼出しを行った場合、コマンドモードにおいてモデム５６から受けたバイトは、計数されない。モデム５６がデータモードに切り換わり、１０バイトが受けられるとすぐに、アプリケーション５４は通知される。

【００４３】モデムがコマンドモードの場合、ユニモデムデバイスドライバは、アプリケーションからのデータＩ／Ｏコマンドを処理しない。しかしながら、ユニモデムデバイスドライバは、ＣＯＭＭ　ＡＰＩからのモデム設定呼出しと、呼制御に関するいくつかの拡張機能を処理することができる。これらの拡張機能は、初期化、ダイアル、および傾聴のような機能を含む。初期化拡張は、モデムを適正設定に初期化する。ユニモデムデバイスドライバは、ＴＡＰＩモジュールに通知することによって初期化が完了した場合、報告する。ダイアル拡張は、ユニモデムデバイスドライバに番号をダイアルすることを要求する。ユニモデムデバイスドライバは、モデムが接続した場合および接続に失敗した場合に報告する。傾聴拡張は、ユニモデムデバイスドライバに到来した呼出しに答えることを要求する。

【００４４】汎用モデムドライバ７４は、多様なモデム形式をサポートすることができる。個々のモデム５６をサポートするために、デバイス特定情報をレジストリにロードする導入サブシステムが存在する。一度この情報がロードされると、汎用モデムドライバは、デバイス特定コマンドを使用してモデムと通信できるようになる。この導入サブシステムについて、図３の参照とともに以下に記述する。

【００４５】図３は、本発明の実施例による導入サブシステムアーキテクチャを説明するブロック図である。導入サブシステムは、コンフィギュレーションマネージャ９０、エニュメレータ９２、クラスインストーラ９４、デバイスローダ、およびレジストリデータ記憶９６を含む。本サブシステムにおいては、上述した仮想デバイスＶＣＯＭＭ７０を、デバイスローダとする。

【００４６】コンフィギュレーションマネージャモジュール９０を使用し、モデムハードウェアのような物理的デバイスと、通信ポートのような論理的装置とを含む本システムにおける通信資源の管理を行う。Windows オペレーティングシステムにおいて、コンフィギュレーションマネージャ９０は、リング０に属する仮想デバイスで

15

ある。特定のローディングの位置および方法が本発明に不足しているとしても、この実現におけるコンフィギュレーションマネージャ９０は、システム起動時にオペレーティングシステムによってロードされる。コンピュータシステム２０をオンにした場合、ＣＰＵは、その起動ルーチンを実行する。この段階の間に、ＣＰＵは、コンフィギュレーションマネージャ９０を含むオペレーティングシステムをロードする。コンフィギュレーションマネージャ９０は、レジストリ９６内の設定情報を含むデータ構造体の情報を読み取ることによって、システム構成について学ぶ。コンフィギュレーションマネージャ９０は、エニュメレータをロードし、コンピュータシステム２０に結合されたモデムデバイスの識別を助ける。コンフィギュレーションマネージャは９０は、資源の競合の検査と、システムにおけるデバイスについての情報の、この場合にはＶＣＯＭＭ７０であるデバイスローダへの報告も行う。コンフィギュレーションマネージャ９０は、新たなデバイスがシステム構成に追加されたことを決定すると、クラスインストーラ９４を呼出し、新たなデバイスについての情報をレジストリ９６にロードする。

【００４７】エニュメレータ９２は、コンピュータシステムに装着されたデバイスを識別するのに使用される検出モジュールである。Windows 95オペレーティングシステムにおいて、エニュメレータ９２は、リング０に属する仮想デバイスである。起動時に、コンフィギュレーションマネージャ９０は、各バスに対して１つのエニュメレータをコンピュータシステム２０にロードする。コンピュータシステム構成に依存して、コンフィギュレーションマネージャは、例えば、ＩＳＡ、ＰＣＭＣＩＡ、ＰＣＩ（Peripheral Component Interconnect 、Peripheral Component Interconnect Special Interest Groupによって支持されるローカルバス標準）、シリアルＣＯＭ、等の、１つまたはそれ以上のエニュメレータをロードすることができる。エニュメレータ９２を起動時に使用し、新たに導入された装置を検出する。ＰＣＭＣＩＡ互換デバイスのようないくつかのモデムは、起動後にロードできることから、エニュメレータ９２を、ランタイムにおいて装着されたモデムデバイスの検出にも使用する。

【００４８】ランタイム前かランタイム中かに装着された、新たに導入されたモデムは、デバイス識別（ＩＤ）によってそれ自身を識別できる。デバイスＩＤを、モデム５６の形式を識別する単一番号または文字列としてもよい。モデム５６が自分自身を識別できる、多くの他の方法が存在する。例えは、モデムは、接続されていることを単に識別し、特別なデバイスＩＤを与えない。このようにした場合、エニュメレータを、デバイス特定情報を読み取り、制御および設定情報がコンピュータシステムに格納されているモデムデバイスを記述するデータを

16

有するこの情報に適合させようとするように設計することができる。

【００４９】いくつかの場合において、導入サブシステムは、モデムに質問してモデム特定情報を発生させ、この情報からデバイスＩＤを得る必要があるかもしれない。この追加のモデム検出について、クラスインストーラ９４と関連して以下に記述する。エニュメレータ９２は、デバイスＩＤを成功裏に決定した場合、このデバイスＩＤをコンフィギュレーションマネージャ９０に報告する。つぎにコンフィギュレーションマネージャ９０は、このデバイスＩＤを有するクラスインストーラを呼出し、その結果、新たなデバイスのデバイス特定情報をレジストリ９６にロードすることができる。

【００５０】レジストリ９６は、コンピュータシステム２０についての情報を保持するデータ構造体である。モデムを導入する状況において、レジストリ９６は、コンピュータシステム２０に結合された１つまたはそれ以上のモデムについての情報を保持する。このモデム情報を見失わないようにするために、レジストリ９６は、ハードウェア接続情報とモデム制御および構成情報とを示す項目の列を含む。モデムがコンピュータシステムに導入されると、これらの項目は、レジストリ９６に追加される。モデム情報を、ユーザがモデム情報を選択および／または入力することを可能にするユーザインタフェースの支援によってレジストリ９６に追加することができる。モデム情報を、クラスインストーラ９４によってレジストリに追加することもできる。

【００５１】本実施例において、導入サブシステムは、デバイスがWindows 95オペレーティングシステムのプラグアンドプレイアーキテクチャに準拠しているかどうかに依存して異なって動作する。非プラグアンドプレイデバイスの場合、ユーザは、導入過程の間、レジストリにロードすべき情報を指定する。これは、レジストリにロードすべきファイルにおけるユーザ指定情報を含んでもよい。ユーザは、ユーザインタフェースにおける制御パネルを通じて、情報を選択および／または入力することができる。ユーザがモデム情報を対話式に選択および入力できるのに加えて、プラグアンドプレイアーキテクチャは、ユーザの介在なしにモデムを導入することができる。プラグアンドプレイデバイスの場合、エニュメレータ９２は、このデバイスがコンピュータシステムにおけるバスに結合されているかどうかを検出する。プラグアンドプレイデバイスを、コンピュータシステムをオンにする前にバスに装着することができ、いくつかの場合において、ランタンムにおいてバスに装着することができる。いずれの場合においても、エニュメレータ９２は、デバイス特定情報をバスから読み出し、デバイスＩＤをコンフィギュレーションマネージャ９０に報告する。コンフィギュレーションマネージャ９０が、モデム５６が新たなものであると決定した場合、クラスインストーラ

９４を呼出し、デバイス特定情報をＩＮＦ（情報）ファイル９８からレジストリにロードする。

【００５２】ＩＮＦファイル９８は、１つまたはそれ以上のモデムに関する特定および制御コマンドを記述する１つまたはそれ以上のスクリプトファイルを含む。ＩＮＦファイルを、コンピュータシステムの補助記憶装置に記録し、必要な場合にアクセスすることができる。モデムスクリプトファイルは、モデムの機能と、モデムの制御に必要なＡＴコマンドとについての記述を含む。クラスインストーラが、モデムに関するＩＮＦファイル９８からの情報を追加した後、ユニモデムデバイスドライバ７４は、レジストリ９６におけるこの情報を、モデムの適切な制御に使用することができる。

【００５３】クラスインストーラ９４は、動的にリンクされたライブラリであり、モデムを識別し、コンピュータシステム２０に導入することに使用することができる。デバイスＩＤを受けるのに応じて、クラスインストーラ９４は、スクリプトファイルから固有情報をレジストリ９６にロードする。クラスインストーラは、ユーザの要求に応じて、モデムスクリプトをレジストリにロードすることができる。代わりに、クラスインストーラ９４は、ユーザの介在なしに、デバイス特定情報をレジストリにロードすることもできる。エニュメレータ９２がモデム５６を認識した場合、コンフィギュレーションマネージャは、デバイスＩＤをクラスインストーラ９４に渡す。次に、クラスインストーラは、固有モデムスクリプトをレジストリに自動的にロードする。

【００５４】エニュメレータがモデムの確認を決定できない場合、クラスインストーラを、モデムの形式の検出に使用することができる。クラスインストーラ９４は、まだ未確認のモデムにコマンド列を発生することによって、モデムに質問する。質問に応じて、モデムは、自分自身についての情報を伝送することによって応答する。モデムを確認するためだけには、多くの情報は利用しないことから、モデムの応答をきれいにし、３２ビット巡回冗長検査を最後に行い、３２ビット数を発生させる。次にクラスインストーラ９４は、この数を使用し、ＩＮＦファイル９８におけるモデムスクリプトに対応する適合するデバイスＩＤを見つける。

【００５５】図４および図５は、モデムの形式を検出する導入サブシステムにおいて実行されるステップを示す。検出方法は、モデムの存在に関して装着されているモデムを有する可能性がある各通信ポートの検査を含む。ポートにおいてモデムが存在しているかどうかを検査するために、クラスインストーラ９４は、最初にＶＣＯＭＭ７０を呼出し、このポートを開く。次にＶＣＯＭＭ７０は、このポートに関するポートドライバ７６をロードし、クラスインストーラ９４とこのポートとの間の通信を確立する。ＶＣＯＭＭ７０を使用して、ポートを開くステップと通信を確立するステップとは、マイクロソフト社から入手できるＶＣＯＭＭ　ＡＰＩに記載されている。

【００５６】通信が一度確立すると、クラスインストーラ９４は、質問を発生し、モデムが接続されている場合、これを決定する（１００，１０２）。この質問は、ＡＴＥ０Ｑ０Ｖ１であり、ＡＴコマンドセットからの標準的なコマンド列である。逐語的にこの質問は、エコーオフ（Ｅ０）、応答する（Ｑ０）、詳しい形式で応答する（Ｖ１）を意味する。

【００５７】クラスインストーラ９４は、標準ボーレートの各々において、初期質問を４回発生する（１０４，１０６）。これらのレートは、９６００、２４００、１２００、および３００を含む。あるボーレートにおいてモデムが検出された場合、クラスインストーラ９４は、次のステップに進む。もしそうでなければ、クラスインストーラ９４は、次の通信ポートに移り、初期質問を繰り返す。

【００５８】モデムが検出された場合、クラスインストーラは、デバイスＩＤを構築しようとする。図４および図５に示す次のステップは、巡回冗長検査プログラムを初期化するステップである（１０８）。このステップにおいて特別に行われる初期化は、決定的なものではない。しかしながら、デバイスＩＤを発生するＣＲＣ計算の前に行うべきである。この実施例において、ＣＲＣ表を瞬時に構築する。この形式のＣＲＣは、ＣＲＣ表の静的なコピーを保持する必要性がなくなるため、性能が向上する。この形式のＣＲＣプログラムの代わりに、種々のＣＲＣプログラムを使用することができるが、この形式のＣＲＣプログラムが好適である。ＣＲＣ表は、瞬時に形成されることから、ＣＲＣプログラムは、この処理の初期に初期化される。ＣＲＣ表を構築する次のステップを、次のステップ１１０として示すが、ステップ１１０は、実際には瞬時に生じることを理解されたい。

【００５９】図４および図５における次の２つのステップ１１２および１１４は、質問リストにおける質問を発生する処理を表す。表１は、リスト上の質問である。

【表１】

質問リスト：

"ATI0\r",	"ATI1\r",	"ATI2\r",	"ATI3\r",
"ATI4\r",	"ATI5\r",	"ATI6\r",	"ATI7\r",
"ATI8\r",	"ATI9\r",	"ATI10\r",	"AT%V\r"

【００６０】表１の各々のコマンドは、モデムに命令し、そのメモリに自分自身についての情報に関して質問させ、その情報を返させる。

【００６１】質問を発生した後、クラスインストーラ

19 20

は、応答を読み取る（１１６）。次にこの応答を、質問の形式に従ってきれいにする。質問がＡＴＩ０の場合（１１８）、第１ラインの第１ワードのみを使用する（１２０）。

【００６２】質問がＡＴＩ４であり、応答文字列が、ヘイズＡＴコマンドセットにおいて示されるＩ４標準に適合する場合、第１ライン全体を使用する（１２４）。

【００６３】他のすべての質問に関して、第１ラインを以下のようにきれいにする。

【００６４】文字列が、以下の表２に示すベイル（bail）リストにある場合（１２６）、ベイルリスト中の文字列に格納されている部分のみを使用する（１２８）。

【表２】

ベイルリスト：

```
"CONNECT",    "RING",    "NO CARRIER",    "NO DIALTONE",
"BUSY",    "NO ANSWER",    "="
```

【００６５】文字列が、以下の表３に示す包含リストにあり、数字に囲まれていない場合（１３０）、文字列全※体を使用する（１３２）。

【表３】

包含リスト：

```
"300",
"1200",
"2400",                                          "2,400",
"9600",        "96",          "9.6",            "9,600",
"12000",       "120",         "12.0",           "12,000",
"14499",       "144",         "14.4",           "14,400",
"16800",       "168",         "16.8",           "16,800",
"19200",       "192",         "19.2",           "19,200",
"21600",       "216",         "21.6",           "21,600",
"24000",       "240",         "24.0",           "24,000",
"26400",       "264",         "26.4",           "26,400",
"28800",       "288",         "28.8",           "28,800",
"31200",       "312",         "31.2",           "31,200",
"33600",       "336",         "33.6",           "33,600",
"36000",       "360",         "36.0",           "36,000",
"38400",       "384",         "38.4",           "38,400",
"9624",        "32bis",       "42bis",          "V32",
"V.32",        "V.FC",        "FAST",           "FAX",
"DATA",        "VOICE"
```

【００６６】文字列が、以下の表４に示す排除リストにある場合、この文字列をスキップする（１３６）。

【表４】

排除リスト：

```
"JAN", "FEB", "MAR", "APR", "MAY", "JUN",
"JUL", "AUG", "SEP", "OCT", "NOV", "DEC"
```

【００６７】上記のステップのいずれもが、第１ラインの文字列に対して満足されない場合、この文字列を次のようにきれいにする。

1） すべての数字を除去する（１３８）。

2） 非６文字に隣接していないすべての６文字を除去する（１４０）。

3） それら自身によってすべての文字を除去する（１４２）。

4） すべてのピリオド、カンマ、および間隔を除去する（１４４）。

【００６８】表１のリスト上のすべての質問を発生した後、巡回冗長検査を、きれいにした応答における各々の文字または数字に関して計算する（１４６）。ＣＲＣの結果は、３２ビット数である。次に３２ビットＣＲＣを、″UNIMODEMxxxxxxxx″の形式のデバイスＩＤを表す文字列に変換する（１４８）。

【００６９】次にクラスインストーラ９４は、デバイス情報がＩＮＦファイル９８に含まれるデバイスＩＤのリストを探索する必要がある。適合するものが見つかった場合（１５０）、クラスインストーラは、適切なスクリプトをレジストリ９６にロードすることができる。

【００７０】しかしながら適合するものが存在しない場合、汎用ＩＤを使用する（１５２）。汎用ＩＤを、モデムサブシステムがモデムが接続されていることを知っているが、モデムを明確に識別できない、および／または適合するデバイスＩＤを見つけられない場合に使用する。汎用ＩＤは、ＩＮＦファイルにおける汎用モデム制御情報に関連する。この汎用モデム制御情報は、広く多種のモデムに大部分適合すると思われるコマンドを基礎とする。汎用制御情報を、接続された特定のモデムに対

して作り変えることができる。さらに、大体のＤＣＥレートを、ＡＴＩ０質問に対する応答に基づいて決定することができる。

【００７１】モデムサブシステムおよび導入サブシステムの双方のアーキテクチャおよび動作を記述したので、例とともに実施例の動作を説明するのに好適になった。

【００７２】この処理の第１段階は、モデムのコンピュータシステムへの導入である。この導入段階は、モデムを認識することと、次にモデム特定情報をレジストリ９６にロードすることとを含む。処理の第２段階は、モデムに依存しないインタフェースを使用し、モデムとのデータＩ／Ｏを制御しかつ行うことを含む。次の例は、ＰＣＭＣＩＡ互換モデムデバイスに関するものである。モデムの形式、すなわち、外付けであるか、内蔵であるかと、含まれる物理的および論理的通信ポートの形式とに依存して、他の導入構成が可能であることを理解されたい。

【００７３】代表的なＰＣＭＣＩＡデバイスに関して、ユーザが、コンピュータシステム２０におけるＰＣＭＣＩＡバスに結合されたＰＣＭＣＩＡカードスロットにＰＣＭＣＩＡカードを挿入したときに、導入は開始する。ユーザは、コンピュータの起動前か、ランタイム中に、デバイスを挿入してもよい。

【００７４】ユーザが、起動前にＰＣＭＣＩＡを導入する場合、起動時にバスにロードされたエニュメレータ９２が、デバイスを検出し、そのデバイスＩＤをコンフィギュレーションマネージャ９０に報告しようとする。ＰＣＭＣＩＡデバイスの場合、ＰＣＭＣＩＡコネクタに装着されたＰＣＭＣＩＡデバイスを検出し、報告するエニュメレータ９２が存在する。同様に、ユーザが、ランタイムにおいてデバイスを挿入する場合、エニュメレータ９２は、同様の検出および報告ステップを行う。エニュメレータ９２が、デバイスを認識できた場合、デバイスＩＤをコンフィギュレーションマネージャに渡す。

【００７５】次に、コンフィギュレーションマネージャ９０は、デバイスＩＤを読み取り、モデムがシステムにおける新たなデバイスであることを認識する。コンフィギュレーションマネージャは、デバイスのリストを格納したレジストリ９６を検査することによって、モデムが新たなデバイスかどうかを決定する。

【００７６】モデムが、エニュメレータ９２に対して自分自身を認識させるようにプログラムされていない場合、クラスインストーラ９４は、デバイスＩＤを発生しようとする。コンフィギュレーションマネージャ９０は、クラスインストーラ９４を呼出し、詳細に上述した検出プログラムを実行する。

【００７７】一度、デバイスＩＤが決定すると、コンフィギュレーションマネージャ９０は、このデバイスＩＤを有するクラスインストーラ９４を呼び出す。クラスインストーラ９４は、このデバイスＩＤに基づく適切なＩＮＦファイル９８を見つけ、ＩＮＦファイル情報をレジストリ９６にコピーする。さらにクラスインストーラ９４は、ＰＣＭＣＩＡモデムを、新たなラインデバイスとしてＴＡＰＩモジュール５８に報告する。

【００７８】レジストリ９６を更新した後、クラスインストーラ９４は、新たなモデム５６が導入されたことを、コンフィギュレーションマネージャ９０に報告する。次に、コンフィギュレーションマネージャ９０は、レジストリ９６を読み取り、デバイスローダを決定する。この実現において、ＶＣＯＭＭ７０はデバイスローダであり、レジストリ９６の入口として認識される（例えば、デバイスローダ＝vcomm.vxd ）。上述したように、デバイスローダとしてＶＣＯＭＭ７０は、適切なドライバのローディングを行い、モデムとの通信を確立する。

【００７９】コンフィギュレーションマネージャ９０は、モデムデバイスと、それに関連する通信資源とを認識するポインタを、デバイスローダ（ＶＣＯＭＭ）７０に与える。明確に、コンフィギュレーションマネージャ９０は、ユニモデムデバイスドライバ７４およびシリアルポートドライバ７０のような、モデムに使用される１つまたはそれ以上のドライバについて、ＶＣＯＭＭ７０に知らせる。この時点で、モデム５６は、導入され、アプリケーションによってアクセスされる準備ができる。

【００８０】アプリケーション５４が、モデム５６との通信を確立したい場合、ＴＡＰＩモジュール５８が、ラインデバイスに質問することによって開始する。ＴＡＰＩモジュール５８は、モデムの認識を、モデムを認識する番号に戻す。次にアプリケーション５４は、″オープン″にすべきモデムのモデム認識を指定するＴＡＰＩモジュール５８に対する呼出しを行うことによって、モデム５６との通信を開く。

【００８１】アプリケーション５４からのこの呼出しに応じて、ＴＡＰＩモジュール６８は、ユニモデムサービスプロバイダ６２を呼出し、モデムデバイスを開く。次にユニモデムサービスプロバイダ６２は、ＶＣＯＭＭ７０を呼出し、ポートを開く。

【００８２】ＶＣＯＭＭ７０が、その名前によってモデムを開く呼出しを受けた場合、ユニモデムデバイスドライバ７４をロードする。ＶＣＯＭＭ７０は、デバイスの名前がレジストリ９６におけるデバイスドライバと関係していることから、ユニモデムデバイスドライバ７４をロードすることを知り、デバイスドライバは、ユニモデムデバイスドライバとして認識される。

【００８３】次に、ユニモデムデバイスドライバ７４は、モデムと関連するポートを開くことを要求する。これに応じて、ＶＣＯＭＭ７０は、適切なポートドライバ（例えば、シリアルポートドライバ、serial.vxd）をロードする。

【００８４】この時点から、アプリケーションは、ＴＡ

ＰＩモジュール５８を経て制御呼出しを行い、モデム５６を制御する（すなわち、番号のダイアル、国内呼出しの待機、等）。ＴＡＰＩモジュール５８は、これらの制御呼出しをユニモデムサービスプロバイダ６２に渡し、このユニモデムサービスプロバイダ６２は、これらの呼出しを、ユニモデムデバイスドライバ７４のより低いレベルの呼出しに変換する。レジストリ９６における情報を使用して、ユニモデムデバイスドライバ７４は、モデム５６を呼出しに対して準備するために適切に初期化する。

【００８５】この時点で、アプリケーション５４は、ＴＡＰＩモジュール５８を呼出し、モデムを介した電話接続を制御することができる。一度、他のモデムとの接続が確立すると、ＴＡＰＩモジュール５８は、この接続についてアプリケーションに通知する。

【００８６】本発明を、特定の実施例の参照とともに記述したが、本発明の範囲から逸脱することなしに、多くの変形例が可能であることを理解されたい。例えば、ＴＡＰＩモジュールおよび通信ドライバの構成を変え、依然としてアプリケーションに対して同様のインタフェースを与えることができる。システムアーキテクチャの多くの特徴は、Windows 95システムの特定のアーキテクチャに基づくものであるが、この特定のアーキテクチャを変更することができることを理解されたい。例えば、他のシステムは、Windows オペレーティングシステムに関して既知の分離した″リング３″および″リング０″を持たなくてもよい。ユニモデムサービスプロバイダおよびユニモデムデバイスドライバを、いくつかのシステムにおいて、結合してもよい。より低いレベルにおいて、ＶＣＯＭＭのようなポートに依存しないインタフェースが存在することは必要なく、同様にユニモデムデバイスドライバを、ＶＣＯＭＭポートドライバとする必要はない。エニミュレータ、ユニモデムデバイスドライバ、およびクラスインストーラによって提供される機能を、異なるモジュールにおいて提供することができ、システムにおいて種々の方法においてロードすることができる。要するに、多くの変形例が可能である。

【図面の簡単な説明】

【図１】本発明の実施例を実現するコンピュータシステムのブロック図である。

【図２】コンピュータシステムにおける本発明の実施例のアーキテクチャを説明するブロック図である。

【図３】本発明の実施例の１つの特徴によるインストーラアーキテクチャを説明するブロック図である。

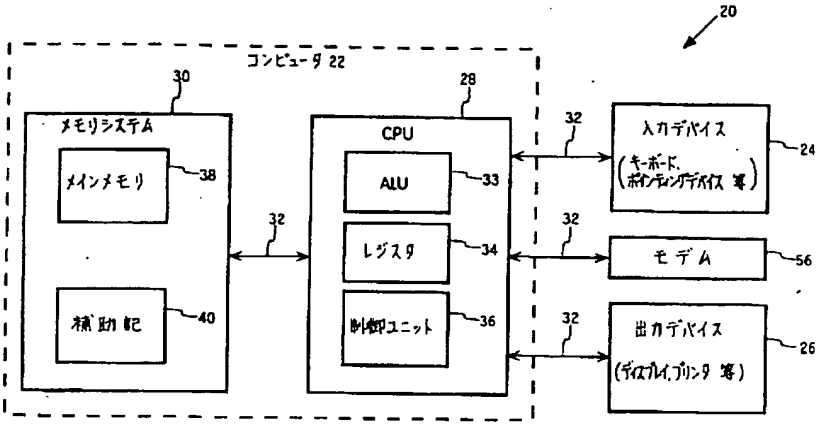
【図４】モデムの形式を検出する導入サブシステムにおいて実行されるステップを説明するフローチャートを図５とともに構成する一部である。

【図５】モデムの形式を検出する導入サブシステムにおいて実行されるステップを説明するフローチャートを図４とともに構成する一部である。

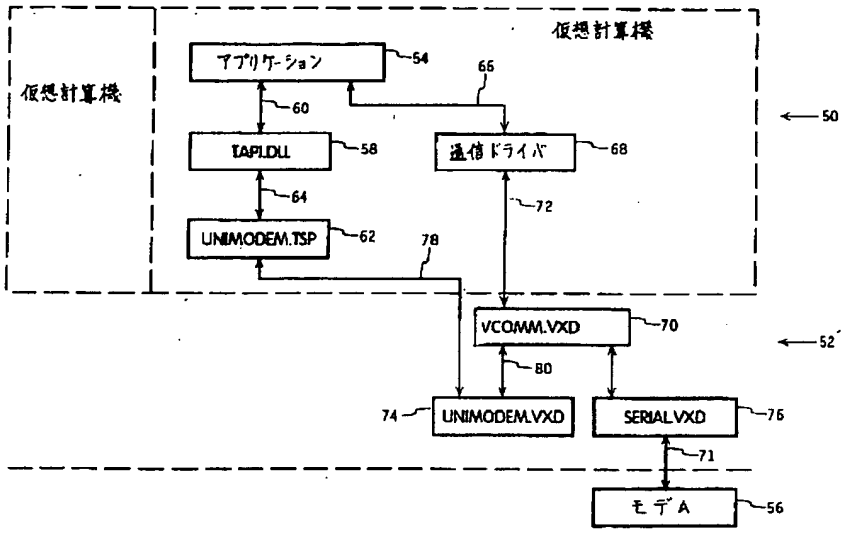
【符号の説明】

２０　コンピュータシステム
２２　コンピュータ
２４　入力デバイス
２６　出力デバイス
２８　中央処理ユニット
３０　メモリシステム
３２　バス構造体
３３　演算器
３４　レジスタ
３６　制御ユニット
３８　メインメモリ
４０　２次記憶装置
５０　リング３
５２　リング０
５４　通信アプリケーション
５６　モデム
５８　ＴＡＰＩモジュール
６０　ＴＡＰＩ
６２　ユニモデムサービスプロバイダ
６４　テレフォニサービスプロバイダインタフェース
６６　通信アプリケーションプログラミングインタフェース
６８　通信ドライバ
７０　仮想通信ドライバ
７１　シリアルポート
７４　汎用モデムデバイスドライバ
７６　シリアルポートドライバ

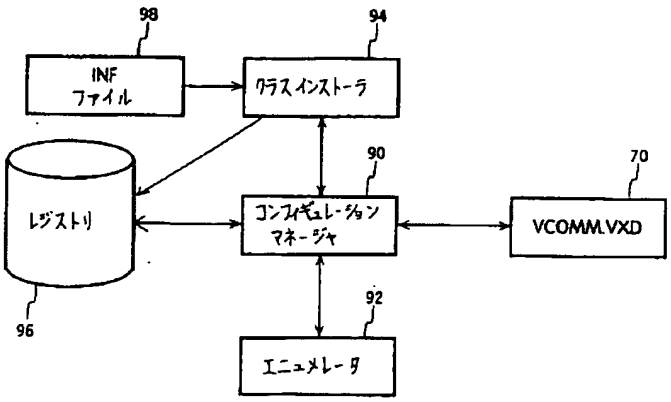
【図1】


20
コンピュータ 22
30
メモリシステム
メインメモリ
38
補助記
40
32
28
CPU
ALU
33
レジスタ
34
制御ユニット
36
32
入力デバイス
(キーボード、ポインティングデバイス 等)
24
32
モデム
56
32
出力デバイス
(ディスプレイ、プリンタ 等)
26

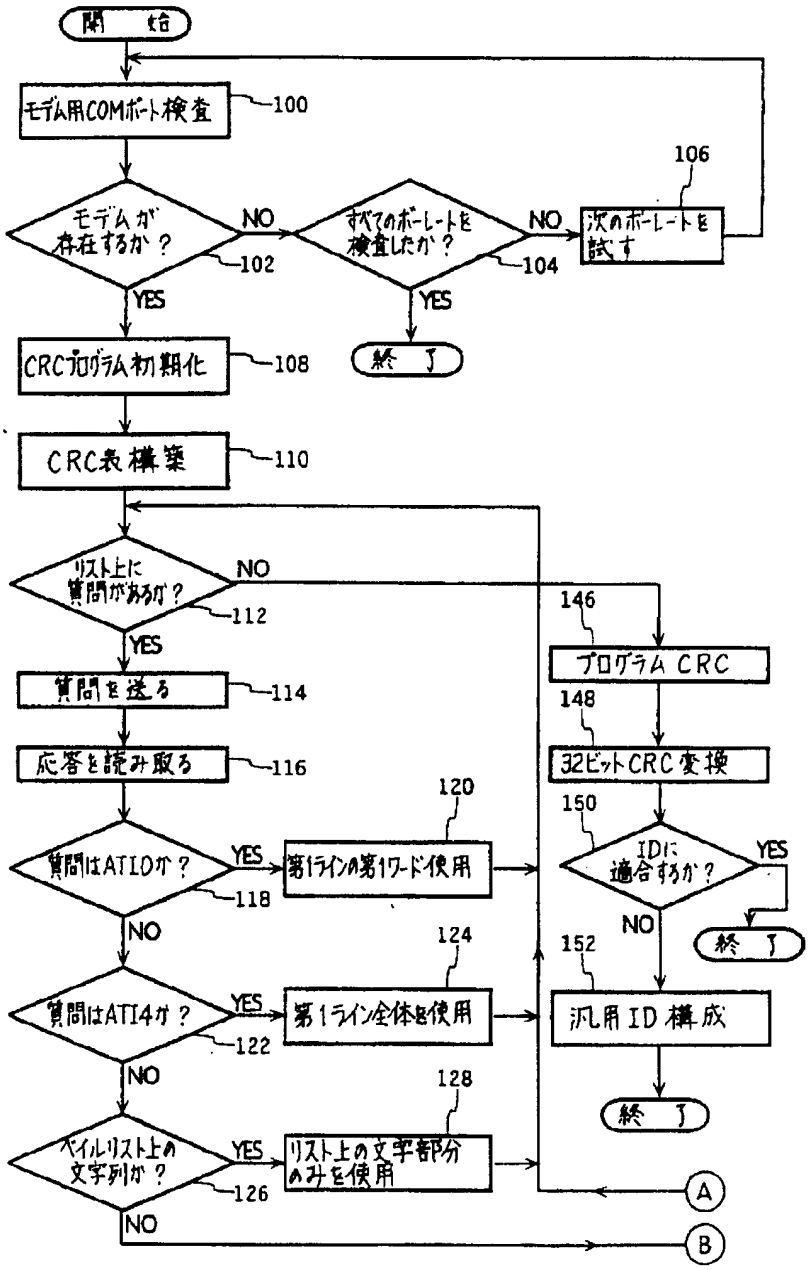
【図2】


仮想計算機
仮想計算機
アプリケーション
54
66
60
TAPI.DLL
58
通信ドライバ
68
64
72
UNIMODEM.TSP
62
78
50
VCOMM.VXD
70
52
80
74
UNIMODEM.VXD
SERIAL.VXD
76
71
モデム
56

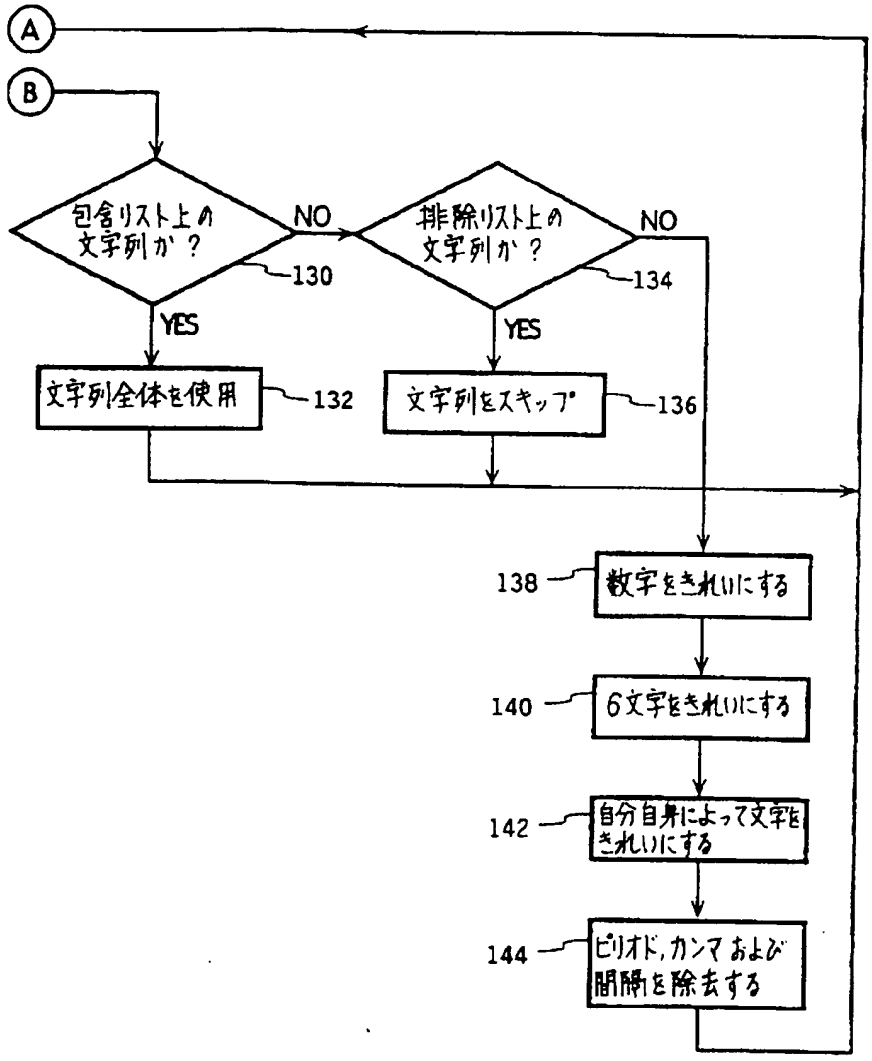
【図3】


98
94
INF
ファイル
クラスインストーラ
90
70
レジストリ
コンフィギュレーション
マネージャ
VCOMM.VXD
96
92
エニュメレータ

【図４】


開　始
モデム用COMポート検査
100
モデムが存在するか？
102
NO
YES
すべてのボーレートを検査したか？
104
NO
YES
106
次のボーレートを試す
終　了
CRC プログラム初期化
108
CRC表構築
110
リスト上に質問があるか？
112
NO
YES
質問を送る
114
応答を読み取る
116
質問はATIDか？
118
YES
120
第1ラインの第1ワード使用
NO
質問はATI4か？
122
YES
124
第1ライン全体を使用
NO
ベイルリスト上の文字列か？
126
YES
128
リスト上の文字部分のみを使用
NO
A
B
146
プログラム CRC
148
32ビットCRC変換
150
IDに適合するか？
YES
NO
終　了
152
汎用 ID 構成
終　了

【図５】



フロントページの続き

(72)発明者　ロバート　ジェイ　ウイリアムス
アメリカ合衆国　ワシントン州　98033
カークランド　ワンハンドレッドトゥエンティエイス　アヴェニュー　エヌイー　10020

(72)発明者　クリストファー　ピー　ガプト
アメリカ合衆国　ワシントン州　98052
レドモンド　ウエスト　レイク　サマミッシュ　パークウェイ　エヌイー　4250　シー－2009

(72)発明者　キース　エイ　リープル
アメリカ合衆国　ワシントン州　98053
レドモンド　エヌイー　セブンティーンスコート　21802